



Rec'd PCT/PTO 24 JAN 2005  
PCT/GB 2003 / 003032



INVESTOR IN PEOPLE

## PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN  
COMPLIANCE WITH RULE 17.1(a) OR (b)

The Patent Office  
Concept House  
Cardiff Road  
Newport  
South Wales  
NP10 8QQ

REC'D 12 AUG 2003

WIPO PCT

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

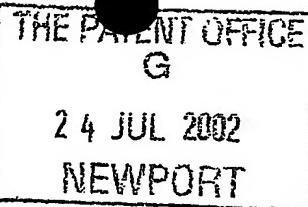
In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 30 July 2003



The  
Patent  
Office

1/77

24JUL02 E735549-1 D00611  
P01/7700 0.00-0217088.4

Patents Act 1977

Rule 16

Request for grant of a patent

The Patent Office  
Concept House  
Cardiff Road  
Newport  
South Wales NP10 8QQ

1. Your reference	GB920020038GB1	124 JUL 2002	
2. Patent application number <i>(The Patent Office will fill in this part)</i>	0217088.4		
3. Full name, address and postcode of the or of each applicant <i>(underline all surnames)</i>	INTERNATIONAL BUSINESS MACHINES CORPORATION Armonk New York 10504 United States of America		
519637001			
If the applicant is a corporate body, give the country/state of its incorporation	State of New York United States of America		
4. Title of the invention	ASYNCHRONOUS MESSAGING IN STORAGE AREA NETWORK		
5. Name of your agent <i>(if you have one)</i>	R J Burt		
"Address for Service" in the United Kingdom to which all correspondance should be sent <i>(including the postcode)</i>	IBM United Kingdom Limited Intellectual Property Department Hursley Park Winchester Hampshire SO21 2JN		
7903925001			
6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and <i>(if you know it)</i> the or each application number	Country	Priority App No <i>(if you know it)</i>	Date of filing <i>(day/month/year)</i>
7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application	No of earlier application		Date of filing <i>(day/month/year)</i>

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? (Answer 'Yes' if:  
 a) any applicant named in part 3 is not an inventor, or  
 b) there is an inventor who is not named as an applicant, or  
 c) any named applicant is a corporate body.)

Yes

9. Enter the number of sheets for any of the following items you are filing with this form.  
 Do not count copies of the same document

Continuation sheets of this form

Description 10

Claim(s) 2

Abstract 1

Drawing(s) 2 *+ 2*

10. If you are also filing any of the following, state how many against each item.

Priority documents

Translations of priority documents

Statement of inventorship and right to grant of a patent (Patents Form 7/77)

Request for preliminary examination and search (Patents Form 9/77)

Request for substantive examination (Patents Form 10/77)

Any other documents  
 (please specify)

11. I/We request the grant of a patent on the basis of this application

  
 Signature  
 R J Burt

 23 July 2002  
 Date

12. Name and daytime telephone number of person to contact in the United Kingdom

 P J Stretton  
 01962 815830

## ASYNCHRONOUS MESSAGING IN STORAGE AREA NETWORK

Field of the Invention

5        This invention relates to systems for asynchronous messaging-and-queuing, and more particularly for the control of storage of messages.

Background of the Invention

10      Asynchronous messaging-and-queuing systems are well known in the art. One such is the IBM MQSeries messaging-and-queuing product. (IBM and MQSeries are trade marks of IBM Corporation.) An MQSeries system is used in the following description, for convenience, but it will be clear to one skilled in the art that the background to the present invention comprises many other messaging-and-queuing systems.

20      In an MQSeries message queuing system, a system program known as a "queue manager" provides message queuing services to a group of applications which use the queue manager to send and receive messages over a network. A number of queue managers may be provided in the network, each servicing one or more applications local to that queue manager. A message sent from one application to another is stored in a message queue maintained by the queue manager local to the receiving application until the receiving application is ready to retrieve it. Applications can retrieve messages from queues maintained by their local queue manager, and can, via the intermediary of their local queue manager, put messages on queues maintained by queue managers throughout the network. An application communicates with its local queue manager via an interface known as the MQI (Message Queue Interface). This defines a set of requests, or "calls", that an application uses to invoke the services of the queue manager. In accordance with the MQI, an application first requests the resources which will be required for performance of a service, and, having received those resources from the queue manager, the application then requests performance of the service specifying the resources to be used. In particular, to invoke any queue manager service, an application first requires a connection to the queue manager. Thus the application first issues a call requesting a connection with the queue manager, and, in response to this call, the queue manager returns a connection handle identifying the connection to be used by the application. The application will then pass this connection handle as an input parameter when making other calls for the duration of the connection. The application also requires an object handle for each object, such as a queue, to be used in performance of the required service. Thus, the application will submit one or more calls

requesting object handles for each object to be used, and appropriate object handles will be dispensed by the queue manager. All object handles supplied by the queue manager are associated with a particular connection handle, a given object handle being supplied for use by a particular connection, and hence for use together with the associated connection handle. After receiving the resources to be used, the application can issue a service request call requesting performance of a service. This call will include the connection handle and the object handle for each object to be used. In the case of retrieving a message from a queue for example, the application issues a "get message" call including its connection handle and the appropriate queue handle dispensed to the application to identify the connection and queue to the queue manager.

With asynchronous messaging systems available today, when a message arrives at a server it is only available to that server, and should that server fail, the message is "trapped" in the server until the server can be restarted.

In high capacity or high performance application architectures the storage of messages in single servers is also a limitation, as a determination has to be made, typically before a message is sent, that the intended destination server is able to handle the message and any subsequent processing required in a timely manner.

There is clearly a need for a more robust and flexible method and system for storage of asynchronous messages in such systems.

#### SUMMARY OF THE INVENTION

The present invention accordingly provides, in a first aspect, a computer system comprising: an asynchronous messaging-and-queuing system; and a storage area network having a storage area network controller; and wherein said storage area network controller comprises control means to control a message queue on behalf of one or more queue managers.

Preferably, said one or more queue managers comprise two or more queue managers, and at least two of said two-or-more-queue-managers are heterogeneous.

Preferably, a message in said message queue is persistent, and wherein said storage area network controller comprises means for controlling persistence of said message.

Preferably, said message is a transactional message, and wherein said storage area network controller comprises transactional control means.

5 Preferably, said transactional control means comprises a syncpoint coordinator.

Preferably, said storage area network controller comprises data integrity control means.

10 Preferably, said data integrity control means comprises a lock manager.

In a second aspect, the present invention provides a method for controlling a computer system having an asynchronous messaging-and-queuing system and a storage area network having a storage area network controller; comprising the steps of: receiving a message request at a queue manager; and passing said message request to said storage area network controller; wherein said storage area network controller comprises control means to control message queues on behalf of one or more queue managers.

20 Preferred method features of the method of the second aspect correspond to the means provided by preferred features of the first aspect.

25 In a third aspect, the present invention provides a computer program to cause a computer system perform computer program steps corresponding to the steps of the method of the second aspect.

30 Using a Storage Area Network (SAN) to hold the message data not only centralizes data storage, it also provides a more robust overall solution, as there is no single point of failure.

35 One definition of SAN is a high-speed network, comparable to a LAN, that allows the establishment of direct connections between storage devices and processors (servers). The SAN can be viewed as an extension to the storage bus concept that enables storage devices and servers to be interconnected using similar elements as in Local Area Networks (LANs) and Wide Area Networks (WANs): routers, hubs, switches and gateways. A SAN can be shared between servers and/or dedicated to one server. It can be local or can be extended over geographical distances.

40 It would be possible, in an embodiment of the present invention, to merely agree a set of protocols for data integrity, transactionality, and other qualities of service between the various cooperating components. In such a case, data integrity, syncpoint coordination, etc. would be

conducted and controlled by a middleware layer, which would supply the appropriate set of primitives to the SAN controller and to the applications and queue managers.

5 By contrast, not only does the presently most preferred embodiment of this invention remove the storage of messages from individual servers and instead store them at the network level, in a SAN, but also provides the support infrastructure in the SAN to supply all required data integrity functionality, allowing multiple queue managers to access the queue (for 10 read and write operations) simultaneously, with complete confidence.

Conventionally, a queue is owned by a specific queue manager, which 15 is responsible for ensuring that multi-threaded access to that queue is maintained in an orderly and correct manner. By moving the queue to the SAN, ownership of the queue is removed from the queue manager and is vested with the SAN controller. Queue managers can apparently access and manipulate messages on the queue as they would a locally owned queue, but the real, underlying management of the manipulation is maintained within the SAN controller.

20 In order for this to work, the SAN Controller may provide the primitives required to control the locking and transactional integrity for the messages on the queue(s) it owns.

25 There are several benefits in the preferred embodiments of the present invention. The first is that messages (data) are removed from the more fragile application server environment into the more robust SAN, where, instead of only being accessible by one server, potentially any server which can connect to the SAN can access the messages.

30 The same benefits cannot be gained simply by mounting the file system holding the queue data, where multiple servers could potentially mount and use the files. If this were to be allowed, conflict situations where, for example, messages locked by one queue manager were deleted by another would 35 rapidly arise, and would make any such system completely unworkable.

By adding locking and two-phase-commit primitives-to-the-SAN. 40 Controller, a preferred embodiment of the present invention allows multiple servers to connect to the SAN and thus simultaneously access the messages on queues (for reads, writes, deletes, locks and transactional operations), with the same level of data integrity that is offered by a single queue manager controlling multi-threaded access to a single queue.

5

A secondary benefit is that it is possible to filter all messages inbound to a particular application to one queue maintained in the SAN. From there they can be distributed to any number of connected servers for subsequent processing by the application with complete transparency to the application.

10

The final main benefit is that since all message data is centrally located, providing for backup and disaster recovery is greatly simplified, as all pertinent data is located in one place, and base SAN services can be utilized to ensure that a secure copy is made.

15

Messages can have the property of being "persistent" - that is they must be logged and journaled by the queue manager before any subsequent processing can occur - or they can be "non-persistent", in which case the message is discarded in the event of a queue manager failure. Preferred embodiments of the present invention are particularly suitable for the control of queues where persistent messages may be placed.

20

The requirement for securing data is the same in a queue controlled by the SAN as it is in a queue locally controlled by a queue manager - that is, authority is required to create and delete a queue, as well as to write and read messages to and from the queue. There are already mechanisms in place (queue clustering) for publishing queue definitions to multiple queue managers, and for providing access control (the local queue manager would determine if access was valid).

25

The SAN Controller would preferably police the connection of queue managers to the SAN, and thereafter assume that a request for queue manipulation sent by a connected queue manager had been validated.

30

Since message data would be flowing over networks, the option to encrypt the data between the SAN and the queue manager would also be a preferred feature.

35

It will be clear to one skilled in the art that the presently preferred embodiment involves the transfer of attributes and activities normally associated with a middleware layer distributed about a networked system into a SAN controller in order to achieve improved robustness, scalability, centralisation of control and ease of maintenance, among other advantages. The attributes and activities associated with middleware are often referred to as "Quality of Service" definitions. It would be possible, as described above, simply to transfer the queue data structures from the local storage of the queue managers into the SAN, and leave the queue managers to negotiate protocols among themselves to manage locking

40

and syncpointing, possibly by means of the conventional middleware provisions. However, as described above, the presently most preferred embodiment of the present invention offers advantages that go beyond those offered by such a solution.

5

As will be clear to one skilled in the art, there will be many other "Quality of Service" definitions that can be incorporated into a SAN controller in the same way as can transactionality, syncpoint coordination, recoverability and so on. One example of such a Quality of Service definition is "Compensability" for subtransactions of a long-running transaction.

10

#### BRIEF DESCRIPTION OF THE DRAWINGS

15

A preferred embodiment of the present invention will now be described by way of example only, with reference to the accompanying drawings, in which:

20

Figure 1 is a block diagram representing the component parts of a system according to a preferred embodiment of the present invention; and

Figure 2 is illustrative of the load-balancing capability of a system according to a preferred embodiment of the present invention.

25

#### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

30

Turning now to Figure 1, there are three main components of presently preferred embodiments of this invention which interact. The first is the SAN (102), controlled by the SAN controller (104); the second is the queue manager (114) which is writing the message to a queue (108) held in the SAN and the third is a queue manager (122) looking to read that message from the SAN held queue (108). Each queue manager (114, 122) is acting on behalf of an application (112, 120) that is making requests that must be satisfied by the queue manager (114, 122). The queue managers (114, 122) and the requesting applications (112, 120) may be located anywhere in a network. That is, systems or system components (110, 118) can be regions or partitions within a system, separate physical computer systems, distributed systems in a network, or any other combination of systems or system components.

35

In particular, to invoke any queue manager service, an application (112, 120) first requires a connection to the queue manager (114, 122). Thus the application (112, 120) first issues a call requesting a connection with the queue manager (114, 122), and, in response to this

40

call, the queue manager returns a connection handle identifying the connection to be used by the application. The application (112, 120) will then pass this connection handle as an input parameter when making other calls for the duration of the connection. The application (112, 120) also requires an object handle for each object, such as a queue (108), to be used in performance of the required service. Thus, the application (112, 120) will submit one or more calls requesting object handles for each object to be used, and appropriate object handles will be dispensed by the queue manager (114, 122). All object handles supplied by the queue manager (114, 122) are associated with a particular connection handle, a given object handle being supplied for use by a particular connection, and hence for use together with the associated connection handle. After receiving the resources to be used, the application (112, 120) can issue a service request call requesting performance of a service. This call will include the connection handle and the object handle for each object to be used. In the case of retrieving a message from a queue (108), for example, the application issues a "get message" call including its connection handle and the appropriate queue handle dispensed to the application to identify the connection and queue (108) to the queue manager (114, 122).

Preferably, the SAN controller (104) of the preferred embodiment of the present invention is provided with a syncpoint coordinator (124), a persistence manager (126) and a lock manager (128). This enables centralization of functions that would otherwise be devolved out to the queue managers, leading to potential problems that may arise in conventional messaging-and-queuing systems.

The preferred embodiment of the present invention is a highly suitable architecture for high throughput systems, with no chance of messages becoming "trapped" in a failed server, and the application throughput can also be "scaled up" by simply connecting more servers to the SAN. Conversely, if demand for the application falls, servers can be disconnected and the maximum possible throughput reduced, on a dynamic basis. As shown in Figure 2, if demand for processing messages in queue (208) rises beyond the capacity of one or more application servers (210), one or more expansion servers (212) can be connected to the SAN, and thus added to the available processing resource available.

Below are described the interactions that may be provided in a presently preferred embodiment of the invention.

**Interaction 1 - Connection**

100 Queue Manager sends connection request to SAN Controller  
105 SAN Controller accepts connection request  
110 SAN Controller verifies identity of Queue Manager  
5 115 If identity confirmed, SAN Controller confirms connection request, else refuses connection

**Interaction 2 - Defining a Queue**

200 Administrator sends a request to define a queue on the SAN  
10 205 SAN Controller validates and if appropriate, accepts request  
210 SAN Controller allocates space for the queue on managed storage  
215 SAN Controller builds necessary control structures  
220 SAN Controller confirms completion of queue creation

**Interaction 3 - Opening a handle to a queue**

300 Queue Manager sends request to open a handle to a queue  
305 SAN Controller confirms existence of queue and authority to open handle  
310 If queue does not exist or incorrect authority, fail the request  
315 SAN Controller opens and returns handle to requesting queue manager  
20 320 SAN Controller updates a usage counter for the queue

**Interaction 4 - Placing a message on the queue**

400 Queue Manager sends a message to place on a queue  
405 SAN Controller verifies authority to place message on queue.  
25 410 SAN Controller writes message data into allocated, managed storage  
415 SAN Controller checks if write is part of syncpoint  
420 If part of syncpoint, SAN Controller places lock on message, confirms to application  
425 If not in syncpoint, SAN Controller confirms message written to queue

**Interaction 5 - Confirming syncpoint (simplified) (read and write****operations)**

500 Queue Manager sends syncpoint confirmation to SAN Controller  
505 SAN Controller confirms queue operation (read or write)  
35 510 SAN Controller clears lock on message, and removes message from queue if read operation

**Interaction 6 - Backing out syncpoint (simplified) (read and write operations)**

40 600 Queue Manager sends syncpoint back out to SAN Controller  
605 SAN Controller confirms queue operation backed out (read or write)  
610 SAN Controller clears lock on message, and removes message from queue if write operation.

Note that any syncpoint operations would typically be of the two phase commit type, but this level of detail is not needed in the present description. Between the SAN Controller and an attached queue manager, a full two phase commit may not be necessary.

5           **Interaction 7 - Reading a message from a queue**

10           700 Queue Manager sends a read request message to SAN Controller

10           705 SAN Controller checks if request is for specific message. If so,  
Interaction 8 - Reading a specific message

15           710 SAN Controller determines next available message to be read

15           715 If not a browse, SAN Controller locks message, and checks if read is  
under syncpoint

15           720 SAN Controller sends message and marks syncpoint if needed

15           725 If read is not a browse and out of syncpoint, message is removed from  
managed storage

20           **Interaction 8 - Reading a specific message from a queue**

20           800 SAN Controller checks if message exists and is not locked by other  
queue manager

25           805 If message is locked or does not exist, read request is rejected

25           810 If not a browse, SAN Controller locks message, and checks if read is  
under syncpoint

25           815 SAN Controller sends message and marks syncpoint if needed

25           820 If read is not a browse and out of syncpoint, message is removed from  
managed storage

30           **Interaction 9 - Closing a handle to a queue**

30           900 Queue Manager sends request to close queue handle

30           905 SAN Controller verifies request and decrements usage counter

30           910 SAN Controller checks the usage counter for the queue

30           912 SAN Controller checks for any uncommitted syncpoints, and if found,  
rejects close handle request

35           915 If usage count is 0, SAN Controller deletes queue handle

35           920 If usage count is not 0, SAN Controller rejects close request

40           **Interaction 10 - Deleting a queue**

40           1000 Administrator sends request to delete queue

40           1005 If request is a "force delete" then delete queue and free allocated  
managed storage

40           1015 SAN Controller verifies that no messages are locked under syncpoint

40           1020 SAN Controller verifies that no other queue managers have open handles

40           1025 If above tests are true, then delete queue and free allocated managed  
storage

1030 If any tests above are false, then reject close request.

**Interaction 11 - Listing owned queues**

1100 Queue manager or system management API sends request to list owned  
5 queues

1105 SAN Controller sends details

**Interaction 12 - Amending queue definition**

1200 Queue manager or system management API sends request to amend queue  
10 definition

1205 SAN Controller verifies request possible and executes changes.

**Interaction 13 - Queue Manager Health Check**

1300 SAN Controller sends health check to each connected queue manager

15 1305 If no response from health check, SAN Controller disconnects failed  
queue manager

**Interaction 14 - Disconnect failed Queue Manager**

1400 SAN Controller terminates each handle owned by the failed queue  
20 manager

1405 SAN Controller checks for all uncommitted syncpoints, and backs them  
out

1410 SAN Controller closes all open handles to queue

1415 SAN Controller closes connection handle to failed queue manager

25 1420 SAN Controller reports failure event

## CLAIMS

1. A computer system comprising:

5 an asynchronous messaging-and-queuing system; and

a storage area network having a storage area network controller; and

10 wherein said storage area network controller comprises control means to control a message queue on behalf of one or more queue managers.

15 2. A computer system as claimed in claim 1, wherein said one or more queue managers comprise two or more queue managers, and at least two of said two or more queue managers are heterogeneous.

20 3. A computer system as claimed in claim 1 or claim 2, wherein a message in said message queue is persistent, and wherein said storage area network controller comprises means for controlling persistence of said message.

25 4. A computer system as claimed in any preceding claim, wherein said message is a transactional message, and wherein said storage area network controller comprises transactional control means.

5. A computer system as claimed in claim 4, wherein said transactional control means comprises a syncpoint coordinator.

30 6. A computer system as claimed in any preceding claim, wherein said storage area network controller comprises data integrity control means.

7. A computer system as claimed in claim 6, wherein said data integrity control means comprises a lock manager.

15 8. A method for controlling a computer system having an asynchronous messaging-and-queuing system and a storage area network having a storage area network controller; comprising the steps of:

receiving a message request at a queue manager; and

0 passing said message request to said storage area network controller;

15 wherein said storage area network controller comprises control means to control message queues on behalf of one or more queue managers.

9. A method as claimed in claim 8, wherein said one or more queue managers comprise two or more queue managers, and said two or more queue managers are heterogeneous.

5 10. A method as claimed in claim 8 or claim 9, wherein a message in said message queue is persistent, and wherein said storage area network controller comprises means for controlling persistence of said message.

10 11. A method as claimed in any of claims 8 to 10, wherein said message is a transactional message, and wherein said storage area network controller comprises transactional control means.

15 12. A method as claimed in claim 11, wherein said transactional control means comprises a syncpoint coordinator.

13. A method as claimed in any preceding claim, wherein said storage area network controller comprises data integrity control means.

20 14. A method as claimed in claim 13, wherein said data integrity control means comprises a lock manager.

15. A computer program comprising computer program code to, when loaded into a computer system and executed, cause said computer system to perform all the steps of a method as claimed in any of claims 8 to 14.

**ABSTRACT****ASYNCHRONOUS MESSAGING IN STORAGE AREA NETWORK**

5       A computer system includes an asynchronous messaging-and-queuing system; and a storage area network having a storage area network controller; and the storage area network controller includes control means to control a message queue on behalf of one or more queue managers, which may be heterogeneous.

10

The storage area network controller may also include means for controlling persistence of messages, transactional control means, such as a syncpoint coordinator, and data integrity control means, such as a lock manager.

15

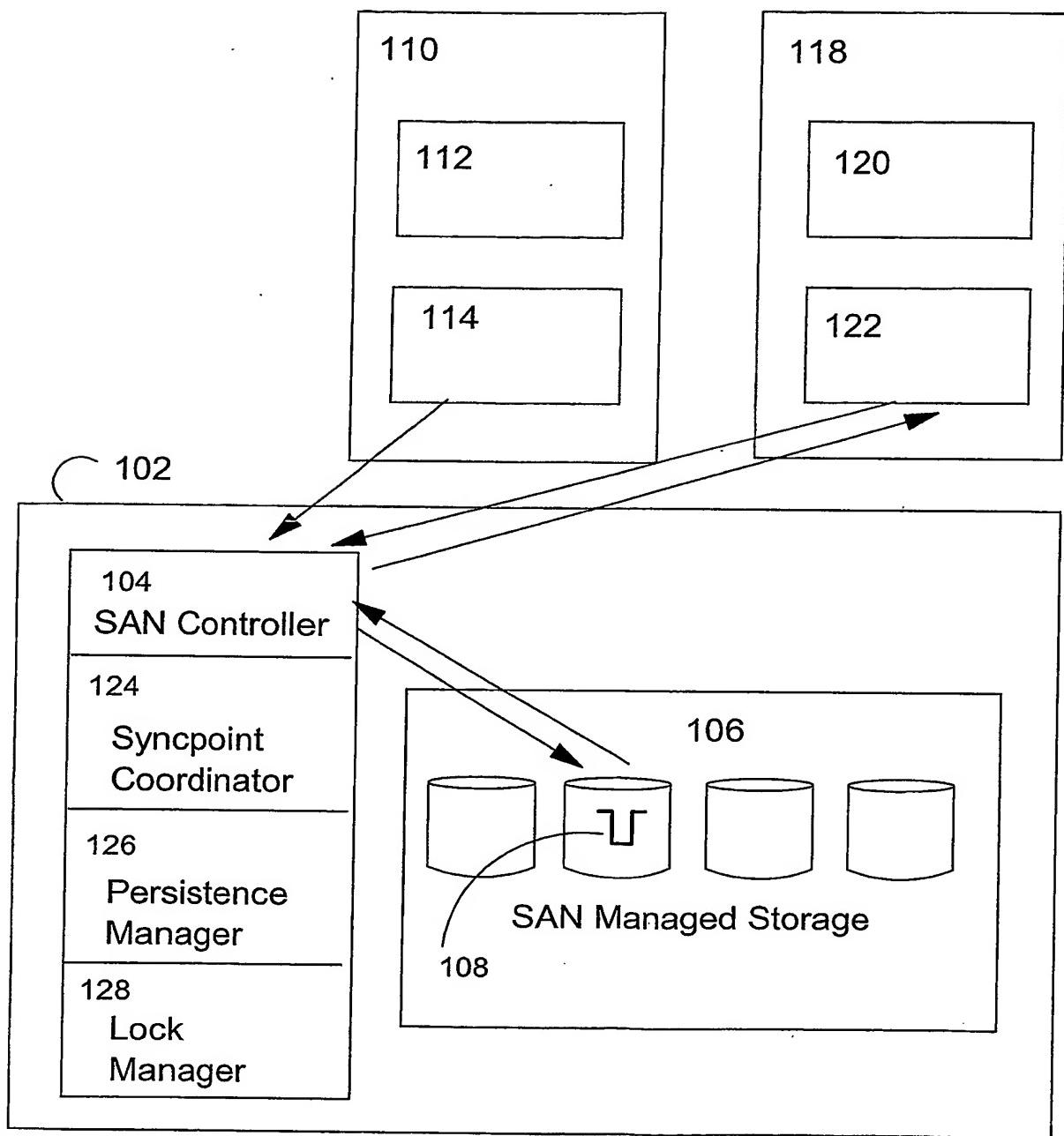


Figure 1

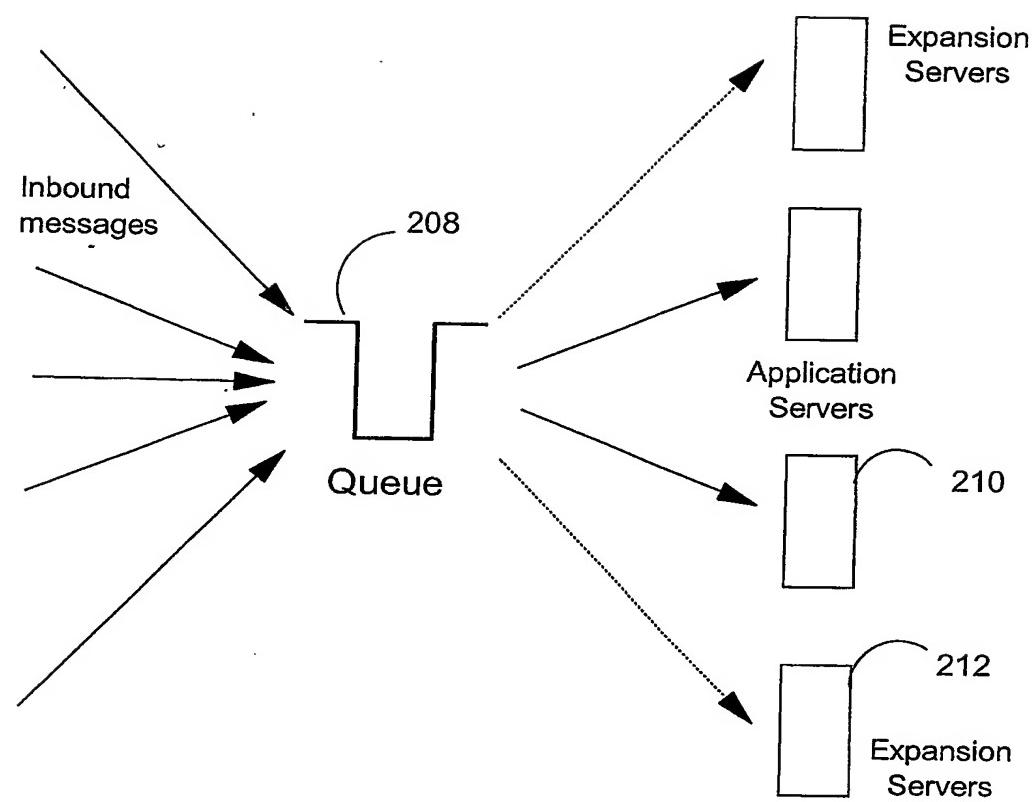


Figure 2

